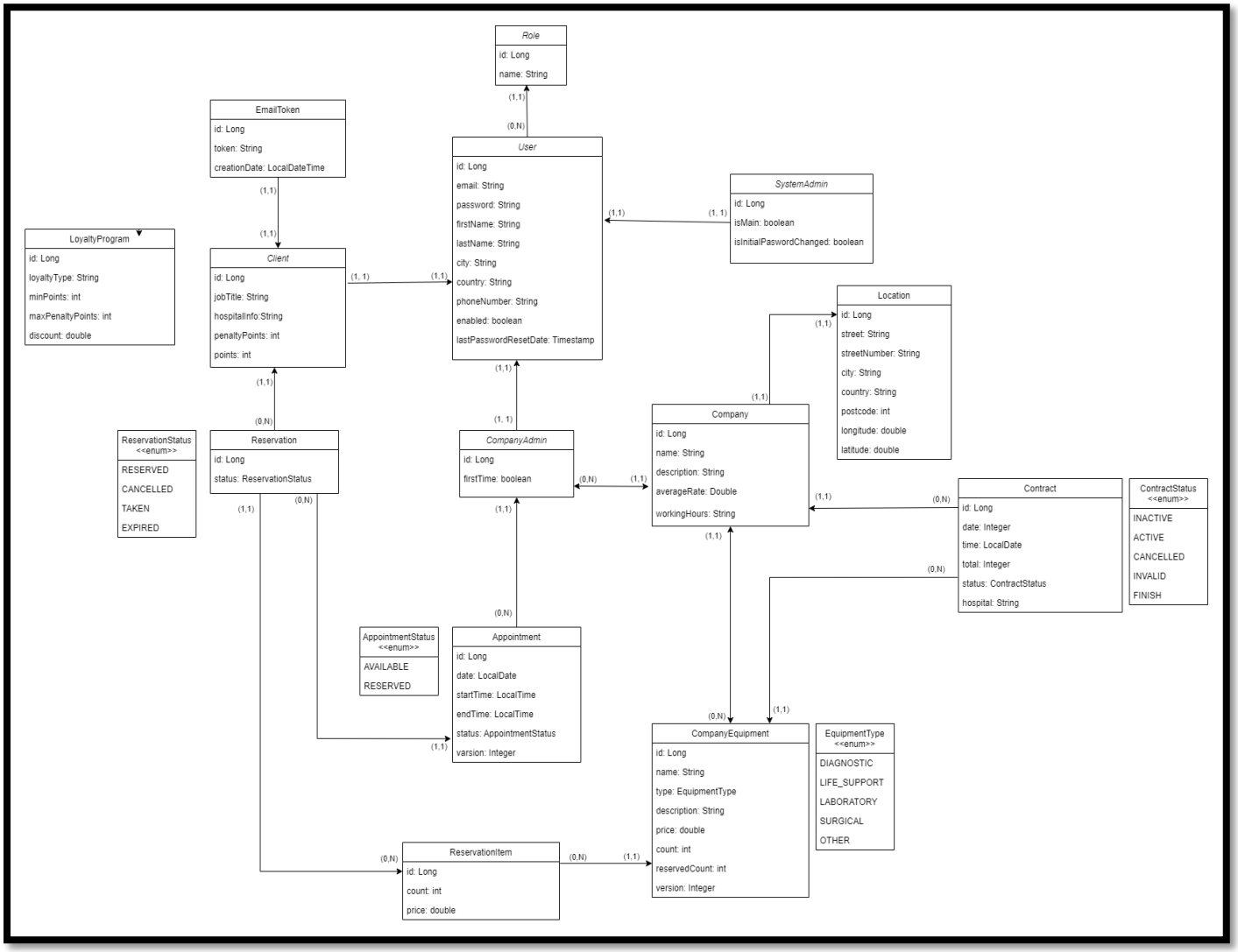
Proof of Concept

Tim 35

1. Klasni dijagram



2. Predlog strategije za particionisanje podataka

U okviru sofisticiranih softverskih rešenja, primena particionisanja podataka igra ključnu ulogu u optimizaciji performansi, povećanju skalabilnosti i jačanju sigurnosti aplikacije.

Jedan od pristupa je horizontalno particionisanje podataka o rezervacijama na osnovu vremena njihovog nastanka. Ova strategija omogućava efikasnije upravljanje informacijama, posebno imajući u vidu da rezervacije iz prošlosti, koje se retko koriste u realnom vremenu, mogu biti smeštene na hardver nižeg kvaliteta. Ovo oslobađa resurse visokog kvaliteta za trenutno relevantne podatke.

Takođe, razmatramo horizontalno particionisanje podataka o terminima, bilo prema datumu ili ID-u kompanije kojoj pripadaju. Ovaj pristup dodatno optimizuje performanse sistema, omogućavajući brži pristup podacima u skladu sa specifičnim potrebama korisnika.

Sigurnosni aspekti su takođe od suštinskog značaja, i tu dolazi do izražaja vertikalno particionisanje podataka o korisnicima. Osetljive informacije poput mailova, korisničkih imena i lozinki čuvamo na posebnim serverima sa pojačanim sigurnosnim merama. Ova vertikalna separacija pruža dodatni sloj zaštite, minimizujući potencijalni rizik od neovlašćenog pristupa ličnim podacima korisnika.

Sve ove strategije particionisanja podataka zajedno doprinose optimizaciji resursa, unapređenju performansi i jačanju bezbednosti, čineći našu aplikaciju efikasnom, skalabilnom i sigurnom.

3. Predlog strategije za replikaciju baze i obezbeđivanje otpornosti na greške

Razmatramo uvođenje slave-master arhitekture kako bismo unapredili performanse i pouzdanost našeg sistema baze podataka. Prema predloženoj arhitekturi, primarna (master) baza biće odgovorna za upisivanje i čuvanje svih podataka, dok će se sekundarne (slave) baze koristiti isključivo za čitanje podataka.

Ovim pristupom postižemo rasterećenje primarne baze od velikog broja zahteva za čitanje podataka, što značajno poboljšava performanse sistema. Takođe, čuvanje kopija podataka u sekundarnim bazama doprinosi boljem očuvanju informacija.

U slučaju da dođe do preopterećenja primarne baze ili do bilo kakvog problema, predviđeno je da jedna od sekundarnih baza preuzme ulogu master baze dok se problem ne reši. S druge strane, ako dođe do pada neke od sekundarnih baza, master baza će preuzeti zahteve te baze dok se ona ne oporavi.

Ova arhitektura obezbeđuje otpornost na greške, poboljšane performanse sistema i sigurnost podataka, jer se opterećenje i odgovornosti ravnomerno raspoređuju između različitih baza. Osim toga, omogućava brzu reakciju na bilo kakav problem, čime se minimizuje potencijalni uticaj na rad sistema.

4. Predlog strategije za keširanje podataka

Uvođenje efikasnog keširanja podataka postaje neophodno u okviru visoko rastućih sistema kako bi se ubrzao pristup informacijama i unapredilo korisničko iskustvo. Visoka frekvencija poziva ka bazi i obimni podaci mogu ozbiljno opteretiti sistem, a time i umanjiti zadovoljstvo korisnika pri korišćenju veb aplikacije.

Strategija keširanja se najbolje primenjuje na podatke koji se ne menjaju često, poput informacija o kompanijama i opremi. Dodatno, keširanje podataka o budućim terminima može se prilagoditi na osnovu analize ponašanja korisnika, smanjujući potrebu za čuvanjem podataka koji nisu odmah relevantni. Na primer, ako se uoči da većina klijenata rezerviše opremu mesec dana unapred, fokusiranje na keširanje podataka o dostupnim terminima
za narednih mesec dana može značajno poboljšati efikasnost sistema.

U slučaju da keš memorija dostigne svoj kapacitet, implementacija LRU (Least Recently Used) strategije omogućava automatsko uklanjanje najstarijih keširanih podataka. Ova taktika obezbeđuje da u kešu uvek budu najaktuelniji i najčešće korišćeni podaci.

Takođe, primena keširanja putem CDN-a (Content Delivery Network) donosi dodatne benefite. CDN omogućava keširanje statičkih fajlova na lokacijama koje su geografski najbliže korisnicima, čime se smanjuje latencija i ubrzava isporuka traženih podataka. Ovaj pristup značajno poboljšava performanse, jer korisnici dobijaju sadržaj mnogo brže nego kada bi svi zahtevi išli direktno na glavni server.

Implementacija ovih strategija keširanja obezbeđuje efikasan i odzivan sistem, pružajući korisnicima brže i zadovoljavajuće iskustvo tokom korišćenja veb aplikacije.

5. Okvirna procena za hardverske resurse potrebne za skladištenje svih podataka u narednih 5 godina

Prosečna veličina jednog red u tabeli, računata spram podataka u skripti za popunu baze(postgres.sql) za entitete od interesa:
User: 171 B
Client: 75 B
Reservation: 55 B
ReservationItem: 63 B
Appointment: 71 B
Company Equipment: 105 B

Napomena: Ovi podaci isključivo zavise od trenutnih podataka i ne uračunavaju indexe i meta podatke koje baza čuva. Bilo bi dobro dodati 10-20% na postojeće vrijednosti.

Procena za narednih 5 godina:
User: 18 GB
Client: 7.5 GB
Reservation: 1.65 GB
ReservationItem: 1.89 GB
Appointment: 2.13 B
Company Equipment: 0.21 GB (uzeli smo svu medicinsku opremu što postoji na svijetu, približno 2 miliona-podaci sa wikipedije)
Ostali entiteti: 10 GB

Stvari koje mogu potencijalno dodatno da utiču su:
-dodavanje slia za korisnike,
-dodavanje više slike za opremu,
-potencijalna unaprijeđenja sistema sa više stavki koje opisuju opremu...

6. Predlog strategije za postavljanje load balansera

Dvije najkorišćenije strategije za postavljenje load balancera u 2022. su: Round-Robin i Least Connections. Odlučili smo se za Least connections, zbog toga što je kod njega manja šansa da se neki server preoptereti(zahtjev prosleđuje serveru sa najmanjim brojem aktivnih konekcija). Uzeli smo ga i zbog činjenice da neki zahtjevi traju duže, kao npr. rezervacije u kojima ima dosta provjera, koje oduzimaju vrijeme.

Least Connections koristimo kada imamo servere u hardverskom smislu iste jačine, a Weighted Least Connection imamo servere različitih serverskih jačina.

Administrator dodaje težine na servere u zavisnosti od njihovih karakteristika, a LoadMaster uzima u obzir kriterijum zadan od strane administratora i broj aktivnih konekcija i na taj način bira kom serveru treba proslijediti zahtjev.

Dodatno:
RateLimiting, koji smo implementirali na primjeru u našoj aplikaciji, je jedan od načina ograničavanja opterećenja servera, takođe je i bitan aspekt bezbjednosti softvera.

7. Predlog koje operacije korisnika treba nadgledati u cilju poboljšanja sistema

Kako bismo napravili aplikaciju koja će svojim ponašanjem da reflektuje personalne potrebe svojih korisnika, treba ubaciti Event Sourcing. Kako su klijenti izbor profita kako za aplikaciju tako i za kompanije, potrebno je prije svega napraviti sistem pretrage i pregleda stranica posjećenih kompanija. Aplikacija bi na osnovu ovih rezultata bila lako proširiva sa preporukama za klijente(potencijalno ubacivanje vještačke inteligencije). Takođe bi admini kompanije imali grafički prikaz koji bi na osnovu datih podataka priložio adekvatan uvid u potražnju određene opreme, a to bi bio značajan izvor podataka za kompanije i korekciju ponude njihove opreme.

Dodatno treba pratiti proces rezervisanja. Koliko često korisnici dođu do stranice za pravljenja rezervacije i odustanu od iste? Koliko često korisnici otkažu prethodno napravljenu rezervaciju? Time bi se dobio bolji uvid u poslovanje.Takođe je bitno praćenje ovih operacija i sa strane performansi jer su to procesi koji su sa kompleksnijom poslovnom logikom, predstavljaju srž aplikacije i traju duže, te težimo da ih optimizujemo.

Neke od 3rd party aplikacija koje mogu pomoći prilikom implementacije i analize događaja jesu Dynatrace, Google analytics, Prometheus...

8. Kompletan crtež dizajna predložene arhitekture